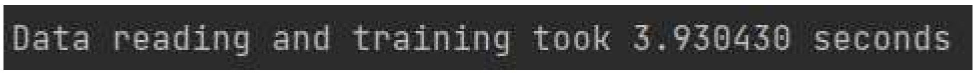
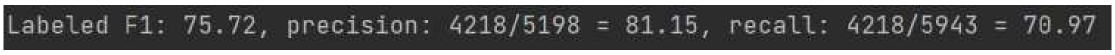
2.1 Part 1: Viterbi Decoding

I implemented the iterative recursion inference and backtracking described in course matrials and the homework handout. The implementation is based on matrix computation.
Running time:



```
Data reading and training took 3.930430 seconds
```

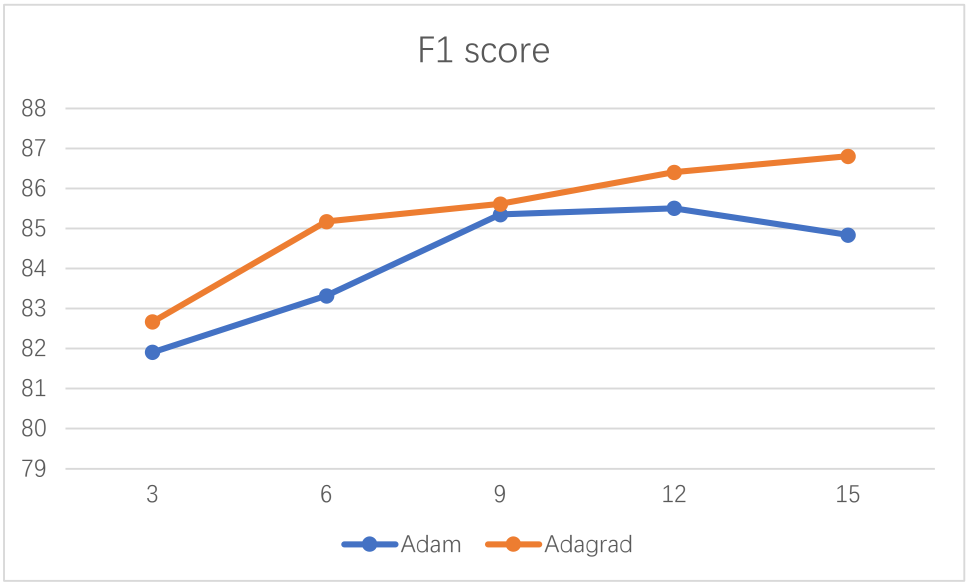
Results:



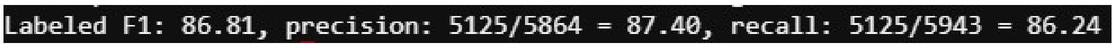
```
Labeled F1: 75.72, precision: 4218/5198 = 81.15, recall: 4218/5943 = 70.97
```

2.2 Part 2: CRF Training

1. For the emssion features, I use the given emission features.
2. For the transition weights, I modify the weights from the HMM model to meet the hard constraints that only B-X or I-X can transition to I-X. For a “to” entry I-X, any weight whose ”from” entry is not B-X or I-X is set to -1000. The weights are attached as “transition.csv”.
3. I implemented two different optimizers, Adam and Adagrad.
The CRF model is trained for 15 epochs. After every 3 training epochs, I save the model parameters and test it on the development set. Here is the F1 score chart:



You can see that using an Adagrad optimizer, after the 15th epoch, my model achieves the best performance on the development set, as shown below:



```
Labeled F1: 86.81, precision: 5125/5864 = 87.40, recall: 5125/5943 = 86.24
```

Thus, I decide to use this model as the final CRF model. The parameters are attached as “model_15.npy”. The CRF's output on the testb blind test set in CoNLL format is attached.

4. I then test the training test. Using the Adagrad optimizer, data reading and training for one

epoch take 116 seconds, as shown in the log “NER_train.o5536283”. Data loading and training for 15 epochs take 1236 seconds, as shown in the log “NER_train.o5536437”.